# CSE 318 : Artificial Intelligence

Offline-2 Max Cut Algorithm (Report)

ID : 2105118

Section : B2

## Introduction

In this offline, we focused on solving the MAX-CUT problem. The goal is to divide the vertices of a graph into two sets such that the total weight of the edges between the sets is as large as possible. Since this problem is very hard to solve exactly, we used different algorithms to find good solutions quickly. We implemented five algorithms and compared how they perform in terms of solution quality and speed.

## Algorithm Overview :

We implemented five algorithms that approach the MAX-CUT problem differently. These include simple methods like random assignments, as well as more advanced strategies that combine randomness and local optimization.

**1. Randomized Algorithm:**

This method randomly assigns each vertex to one of the two sets. It is extremely fast and simple, but the solution quality is usually poor. This method is mainly used as a baseline to compare against more intelligent strategies.

**2. Greedy Algorithm:**

The greedy approach starts with an empty partition and adds each vertex to the set that increases the cut the most. It is a straightforward method and performs much better than random, but it can make early decisions that hurt the overall solution quality.

**3. Semi-Greedy Algorithm:**

This method improves on the greedy strategy by using a parameter called alpha. Instead of always picking the best vertex, it builds a short list of good candidates and randomly chooses one from the list. This adds some randomness and prevents the algorithm from getting stuck in poor choices made early.

**4. Local Search:**

Once a full solution is built, local search improves it by looking for vertices that can be moved to the other set to increase the cut value. It repeats this process until no better moves are found. It is a powerful way to enhance any initial solution.

**5. GRASP (Greedy Randomized Adaptive Search Procedure):**

GRASP combines the Semi-Greedy method with Local Search. It repeatedly builds random-but-good solutions and improves them using local search, keeping the best

one found. This method often gives the best results but takes more time because it runs several rounds.

## Performance Comparison :

| Name | Pros | Cons |
|---|---|---|
| Randomized | Very fast | gives the worst results |
| Greedy | Fast and better than random | limited by early choices |
| Semi-Greedy | Fast and more flexible | gives better solutions than greedy |
| Local Search | Improves any given solution | Slower |
| GRASP | gives the best results overall | Slower because it repeats the process many times |

Overall, combining different strategies like in GRASP leads to the most reliable outcomes.